What is claimed is:

1. A method for balancing the load of a parallel processing system having a plurality of parallel processing elements linked serially in a line with first and second ends, wherein each of said plurality of processing elements has a local number of tasks associated therewith, the method comprising:

determining a total number of tasks present on said line;

notifying each of said plurality of processing elements of said total number of tasks;

calculating a local mean number of tasks for each of said plurality of processing elements;

calculating a local deviation for each of said plurality of processing elements;

determining a first local cumulative deviation for each of said plurality of processing elements;

determining a second local cumulative deviation for each of said plurality of processing elements; and

redistributing tasks among said plurality of processing elements in response to said first local cumulative deviation and said second local cumulative deviation.

2. The method of claim 1 wherein said determining a total number of tasks present on said line comprises sequentially summing said local number of tasks associated with each of said plurality of PE's from said first end of said line to said second end of said line.

3. The method of claim 1 wherein said determining said total number of tasks present on said line includes solving the equation $V = \sum_{i=0}^{i=N-1} v_i$, where V represents said total number of tasks present on said line, N represents the number of processing elements in said line, and v_i represents said local number of tasks associated with a local PE_r.

4. The method of claim 1 wherein said notifying step includes passing said total number of tasks from said second end to said first end.

5. The method of claim 1 wherein said calculating a local mean number of tasks for each of said plurality of processing elements further includes solving the equation $M_r = Trunc((V + E_r) / N)$, where M_r represents said local mean for PE_r, N represents the total number of processing elements in said line, and E_r is a number in the range of 0 to $(N - 1)$.

6. The method of claim 5 wherein each PE has a different E_r value.

7. The method of claim 5 wherein E_r controls said *Trunc* function such that said total number of tasks for said line is equal to the sum of the local mean number of tasks for each of said plurality of processing elements in said line.

8. The method of claim 5 wherein said local mean $M_r = Trunc((V + E_r)/N)$ for each local PE_r on said line is equal to one of X and $(X+1)$.

9. The method of claim 1 wherein said calculating a local deviation for each of said plurality of processing elements includes finding the difference between said local number of tasks and said local mean number of tasks for each PE_r.

10. The method of claim 1 wherein said determining a first local cumulative deviation includes sequentially summing said local deviations for each PE_r from said first end of said line to adjacent upstream PE_{r-1} on said line.

11. The method of claim 1 wherein said determining a second local cumulative deviation includes finding a difference between the negative of said local deviation and said first local cumulative deviation for each PE_r.

12. The method of claim 1 wherein said redistributing tasks among said plurality of processing elements comprises:

transferring a task from a local PE_r to a left-adjacent PE_{r-1} if said first local cumulative deviation for said local PE_r is a negative value;

transferring a task from said local PE_r to a right-adjacent PE_{r+1} if said second local cumulative deviation for said local PE_r is a negative value.

13. The method of claim 1 wherein said redistributing tasks among said plurality of processing elements comprises:

transferring a task from a local PE_r to a left-adjacent PE_{r-1} if said second local cumulative deviation for said local PE_r is a positive value;

transferring a task from said local PE_r to a right-adjacent PE_{r+1} if said first local cumulative deviation for said local PE_r is a positive value.

14. The method of claim 1 wherein said calculating a local mean number of tasks for each of said plurality of processing elements; said calculating a local deviation from said local mean number of tasks for each of said plurality of processing elements; said determining a first local cumulative deviation; said determining a second local cumulative deviation; and said redistributing tasks are completed in parallel for each of said plurality of processing elements.

15. The method of claim 1 wherein said calculating a local deviation for each of said plurality of processing elements, determining a first local cumulative deviation for each of said plurality of processing elements, determining a second local cumulative deviation for each of said plurality of processing elements, and redistributing tasks among said plurality of processing elements in response to said first local cumulative deviation and said second local cumulative deviation are repeated until said local deviation, said first local cumulative deviation, and said second local cumulative deviation for each of said plurality of processing elements is zero.

16. A method for assigning tasks associated with a parallel processing element (PE_r), within a line of processing elements, said PE_r being serially connected to at least one other PE in said line and said PE_r having a local number of tasks associated therewith, the method comprising:

notifying said PE_r of a total number of tasks associated with said line;

determining said PE_r's associated share of said total number of tasks;

determining a first local cumulative deviation for said PE_r;

determining a second local cumulative deviation for said PE_r;

redistributing tasks with said at least one other connected parallel processing element in response to at least one of said first local cumulative deviation and said second local cumulative deviation.

17. The method of claim 16 wherein said notifying said PE_r comprises:

serially summing said local number of tasks present on said line; and

transmitting said total number of tasks to said PE_r.

18. The method of claim 16 wherein said determining said PE_r's associated share of said total number of tasks comprises:

calculating a local mean number of tasks for said PE_r, said local mean number of tasks being equal to $M_r = Trunc((V + E_r) / N)$, where M_r represents said local

mean of said local PE_r, N represents the total number of processing elements in said line, and E_r represents a number in the range of 0 to $(N-1)$; and

calculating a local deviation from said local mean number of tasks for PE_r by finding the difference between said local number of tasks and said local mean number of tasks for PE_r.

19. The method of claim18 wherein each processing element in said line has a different E_r value.

20. The method of claim 16 wherein E_r controls said *Trunc* function such that said total number of tasks for said line is equal to the sum of the local mean number of tasks for each of said processing elements in said line.

21. The method of claim 16 wherein $M_r = Trunc((V + E_r) / N)$ for each local PE_r on said line is equal to one of X and $(X+1)$.

22. The method of claim 16 wherein said determining a first local cumulative deviation for PE_r includes summing said local deviations for each upstream PE within said line.

23. The method of claim 16 wherein said determining a second local cumulative deviation for said PE_r includes finding the difference between the negative of said local deviation and said first local cumulative deviation for said PE_r.

24. The method of claim 16 wherein said redistributing tasks with said at least one other parallel processing element comprises:

transferring a task to said at least one other parallel processing element from said PE_r if at least one said first local cumulative deviation is a negative value and if said second local cumulative deviation is a negative value; and

transferring a task from said at least one other parallel processing element to said PE_r if at least one said first local cumulative deviation is a positive value and if said second local cumulative deviation is a positive value.

25. The method of claim 16 wherein said determining a first local cumulative deviation step, determining a second local cumulative deviation step, and said redistributing tasks step continue until said first local cumulative deviation and said second local cumulative deviation for said PE_r is zero.

26. A memory device carrying a set of instructions which, when executed, perform a method comprising:

determining a total number of tasks present on said line;

notifying each of said plurality of processing elements of said total number of tasks;

calculating a local mean number of tasks for each of said plurality of processing elements;

calculating a local deviation for each of said plurality of processing elements;

determining a first local cumulative deviation for each of said plurality of processing elements;

determining a second local cumulative deviation for each of said plurality of processing elements; and

redistributing tasks among said plurality of processing elements in response to said first local cumulative deviation and said second local cumulative deviation.